M-014

タイマを用いる分散システムにおける時間因果順序違反を保存した障害復旧方法

A fault recovering method for distributed system with timer preserving timed causal order

松本 真治 †　　樋口 昌宏 †
Shinji Matsumoto　　Masahiro Higuchi

1. 目的

本研究ではタイマを用いる分散システムにおけるスナップショットの取得とロールバック方法、ロールバック時に起きる問題点と解決方法の検討を行った。

2. タイマを用いる分散システム

本研究で対象とするシステムはタイマと連携して動作するプロセスがネットワークを介してメッセージをお互いに送受信することで協調動作を行うものである。各プロセスは複数のタイマを起動することができ、どのタイマがタイムアウトしたかはタイムアウト信号の種類により区別することができるものとする。

そのようなシステムでは、各プロセスはタイマを用いて通信相手を監視をする。各プロセスは自身の発行する要求メッセージに対するレスポンス時間 T_{rsp} を算定しておき実際に要求メッセージを送信の際に $T_{rsp} \times \alpha$ ($\alpha > 1$) のタイマを起動し、そのタイマがタイムアウトすると相手がレスポンスを返さないものと判断する。例えば、要求がサーバに対するものであればタイムアウトにより当該サーバがダウンしたものと考え他のサーバを探す。ネットオークションではタイムアウトまでに応答しないプロセスは応札する意思がないと判断する。

また、種々の理由によるロールバックを可能とするために頻繁にスナップショットを取得する場合を考える。

3. 一般的なスナップショット取得方法

分散システムにおいて大域状態は各プロセスの状態と各通信リンク中のメッセージ系列からなる。分散システムにおけるスナップショットは分散システムのある実行における因果順序に矛盾しない切断を表す大域状態のことをいう。スナップショットを随時取得しておくことによりシステム障害時にスナップショットの状態から再開 (ロールバック) することで障害復旧が可能となる。スナップショットの取得は (1A 始動プロセスの場合) スナップショットを起動する,(1B 受動プロセスの場合) マーカーを受信する,(2) マーカーを送信する, (3) 自身の状態を記録する,(4) 各入力チャンネルについてマーカーを受信するまでに受信したメッセージ系列を記録する,(5) すべての入力チャネルからマーカーを受信すると終了, という手順で行う。[1]

4. タイマに関する情報の保存とロールバック

タイマを用いる分散システムでは、タイマの状態を大域状態に含める必要がある。しかし、タイマは時間を計測することしかできないのでそれを記録することができない。そこでタイマ管理プロセスというものを導入することを考えた。記録しなければならないタイマの状態は (1) どのタイマが起動中か,(2) 停止命令を受信しない場合それらがどの順にタイムアウトするか, の２つであるので、図１のようにタイマ管理プロセスがプロセスとタイマ間の信号を仲介することで常にどのタイマが動作中かを把握する。スナップショットを取得するときには、その後のタイマの動作を監視し、動作中であったタイマがどの順にタイムアウトしたかを記録する。このためスナップショット取得中にタイマへの停止命令が発行されてもこれをタイマには伝えず、さらに停止命令が発行されたタイマがタイムアウトしても、これをプロセスに伝えない。1 つのみを残してすべてのタイマがタイムアウトするとスナップショットの取得は終了となる。

ロールバック時にはタイマ管理プロセスが記録通りにタイムアウト信号を順次プロセスに伝える。以上により、タイマを用いるシステムでも障害時に整合性を持つ大域状態へのロールバックが可能となる。

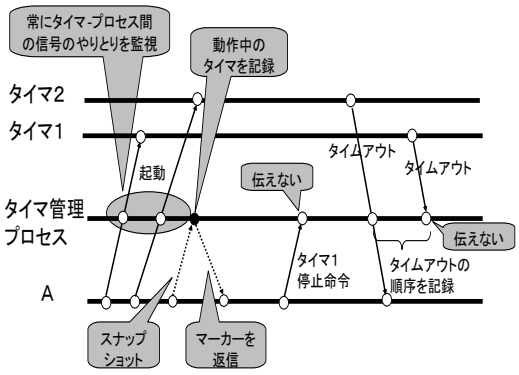

図 1　タイマ管理プロセスの役割

5. タイムアウト信号の発生方法と問題点

5.1 タイムアウト信号

4 で述べた方法ではタイマの情報としてタイムアウトの順序しか考慮していないため、例えばロールバック時に初めにタイムアウトを一斉発生させ、その後でメッセージを随時配送するといったことも可能となってしまう。このような場合 2 で述べたようにサーバがダウンしたものと考えられてしまったり、オークションでは応札する意思がないと判断されてしまうことを意味する。実際にはその後にサーバからの応答やオークションでの応札メッセージを受

† 近畿大学 大学院 総合理工学研究科 エレクトロニクス系工学専攻

信することも充分考えられ、望ましいロールバック方法とは言えない。そこで、タイムアウトはなるべく後に発生させる方法を考えた。

5.2 ロールバック方法

タイムアウト信号をなるべく後で発生させるロールバック方法として以下のものを考えた。(1) 記録しているタイマの計測時間 t に対応した計測時間 (例えば t/2) を設定したタイマを起動する, (2) 記録しているメッセージの受信イベントを復活させる,(3)(1) で起動したタイマがタイムアウトするとこれをプロセスに伝える。しかし、この方法では次に述べる問題が生じてしまう。そこで、その問題点と解決策を以下に示した。

5.3 時間因果順序違反

本研究ではメッセージとタイムアウト信号の順序関係も考慮し、これを時間因果順序として定式化した。

前節で述べた方法はロールバックする大域状態では時間の量的な情報が失われており、タイムアウトをなるべく後で発生させる方法では時間因果順序違反という問題が起こる可能性がある。

ここで因果順序を $\prec$, と表すとする。上記のようなロールバックを行った場合、図 2 のように「タイマ A の起動 $\prec$ タイマ B の起動」,「タイマ A の計測時間 $<$ タイマ B の計測時間」であれば、本来「タイマ A のタイムアウト時刻 $<$ タイマ B のタイムアウト時刻」という関係が成り立つはずであるにもかかわらず「タイマ B のタイムアウト $\prec$ タイマ A のタイムアウト」という順序でイベントが発生してしまう可能性が考えられる。通常このような動作はプログラマが想定しないものと考えられる。このような場合を時間因果順序違反と呼ぶ。

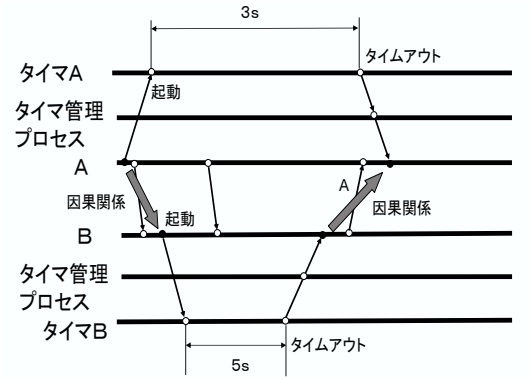

図 2　時間因果順序違反例

5.4 解決策

分散システムにおいて論理クロックを用いて因果順序メッセージ配送を実現する方法が提案されている。ここで考える方法でもメッセージ配送はそれに基づいて因果順序メッセージ配送を行う。

時間因果順序違反を回避するため図 3 のようにタイマ管理プロセスにメッセージ配送にも介入させる方法を考える。タイマ管理プロセスは { タイマの起動論理時刻、計測時間、タイムアウト時の論理時刻 } からなるタイマレコードを管理する。さらにメッセージを他プロセスに送信する際タイマレコードを付加することによりタイマ管理プロセス間でタイマの動作情報を共有する。図 3 ではスナップショット取得時にタイマ A のタイマ管理プロセスは {①, τ_A , not yet} というタイマレコードを持っている。タイマ B のタイマ管理プロセスはこれに加えて {②, 5s ,③} というタイマレコードを持っている。「① $\prec$ ②」なので、さらに「τ_A $<$ 5s」ならばタイマ A のタイムアウトはタイマ B のタイムアウトより早く起こるはずである。しかし、ロールバック後は時間の情報が失われていることからタイマ A のタイムアウト信号が発生する前に ④ で B からのメッセージを受信してしまうことも起こるうる。この場合、ただちにメッセージを配送すると「③ $\prec$ ⑤」となってしまい時間因果順序違反となる。しかし、そのメッセージに付加されているタイマレコード {②, 5s ,③} を見ることにより自身の持っているタイマレコード {①, τ_A ,not yet} と比較し、「① $\prec$ ②」、「τ_A $<$ 5s」のことからタイマ A のタイムアウトが先に発生するべきと判断しメッセージの配送をタイムアウト信号の後の ⑥ までメッセージを遅らせて違反を解消する。

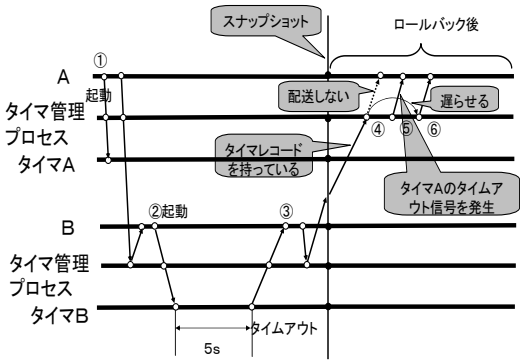

図 3　時間因果順序違反の回避

5.5 タイマレコードの削除

5.4 のままだと不必要なタイマレコードがたまってしまうので、タイマレコードに”各プロセスへのタイムアウト伝達状況”を追加し、タイマレコードの状態を { タイマの起動論理時刻、計測時間、タイムアウト時の論理時刻、各プロセスへのタイムアウト伝達状況 } とする。そして、あるタイマがタイムアウトしたことが分散システム上のすべてのプロセスに伝わるとタイマレコードを削除する。こうすることによって、不必要なタイマレコードがたまっていくのを防ぐことができる。

6. 今後の予定

今回考慮したシステムに対してのシミュレートプログラムを作成し今回示したタイムアウトをなるべく起こさないロールバック方法の有用性、ロールバックアルゴリズムの正当性の検証。各プロセスに無茶な条件を強制的に与え通常のときと変わらずに動作するのか等のデータを採取していく予定である。

参考文献

[1] Venkatesh, A.K. et al.; ”Optimal chckpointing and local recording for domino free rollback recovery,” Information Processing Letters 25(1987),295-303.